Connect mysql database to shindig

This tutorial shows how one can add a database to Java Shindig, so that social requests to shindig will be taking data from this database. For this tutorial MySQL database is used.

1) Download shindig trunk as follows:

```bash
$mkdir shindig
$cd shindig
# This also was working for a branch 2.0.x
# http://svn.apache.org/repos/asf/shindig/branches/2.0.x/
```

2) To use database in Shindig one could take advantage of existing in shindig "samples" project that uses JPA and EclipseLink to connect to the database from Java. This how Shindig should be compiled in this tutorial (in future it should be cleaned up to have only one command for compilation)

```bash
# Compilation: in shindig root folder
# To build samples jar
$mvn -Dmaven.test.skip -P samples
# To build server jar
$mvn -Dmaven.test.skip
# Running server
$cd java/server
$mvn jetty:run
```

3) First of all we need to add JPASocialModule.java into Shindig's Guice modules. This module will provide real database implementations for PersonService, AppDataService, etc. One should add this module in java/server/src/main/webapp/WEB-INF/web.xml. Some default Guice modules should be removed from this file in order to avoid compilation problems. This is how the <context-params> section should look.

```xml
<context-param>
  <param-name>guice-modules</param-name>
  <param-value>
    org.apache.shindig.common.PropertiesModule:
    org.apache.shindig.gadgets.DefaultGuiceModule:
    org.apache.shindig.social.core.config.SocialApiGuiceModule:
    org.apache.shindig.social.opensocial.jpa.spi.JPASocialModule:
    org.apache.shindig.gadgets.oauth.OAuthModule:
    org.apache.shindig.common.cache.ehcache.EhCacheModule:
  </param-value>
</context-param>
```

We also need to change the SocialApiGuiceModule.java to add a sample OAuthDataStore.
import org.apache.shindig.social.opensocial.oauth.OAuthDataStore;
import org.apache.shindig.social.sample.oauth.SampleOAuthDataStore;

protected void configure() {
    ....

    bind(BeanConverter.class).annotatedWith(Names.named("shindig.bean.converter.atom"))
        .to(BeanXStreamAtomConverter.class);

    bind(OAuthDataStore.class).to(SampleOAuthDataStore.class);

    ....
}

4) Here we will adjust Shindig settings to compile with database support. We have to move external dependencies from samples/pom.xml to social-api/pom.xml. We also add dependency on shindig-samples and (since we use mysql database) we add dependency on the database driver too. Now to compile the project we have to compile first "samples" and then the whole project (as explained in 2)

    <!-- dependency on shindig-samples -->
    <dependency>
        <groupId>org.apache.shindig</groupId>
        <artifactId>shindig-samples</artifactId>
        <version>${project.version}</version>
    </dependency>

    <!-- add a driver for mysql -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
    </dependency>

    <!-- add here all external dependencies from samples/pom.xml -->

5) Generate signing keys for OAuth with the following commands. (In production use the proper keys)

    $mkdir ssl_keys
    $cd ssl_keys
    $openssl req -newkey rsa:1024 -days 365 -nodes -x509 -keyout testkey.pem
        -out testkey.pem -subj '/CN=mytestkey'
    $openssl pkcs8 -in testkey.pem -out oauthkey.pem -topk8 -nocrypt -outform
        PEM

Add the keys information into java/common/conf/shindig.properties. Don't forget the full path to your oauthkey.pem!!

    shindig.signing.key-name=mytestkey
    shindig.signing.key-file=/path_to_shindig_branch/ssl_keys/oauthkey.pem

6) Add your database information to java/samples/src/main/resources/socialjpa.properties. Here we use mysql database, so the settings are
mysql specific.

```
db.driver=com.mysql.jdbc.Driver
db.url=jdbc:mysql://localhost:3306/shindig
db.user=shindig
db.password=shindig
db.write.min=1
db.read.min=1
jpa.socialapi.unitname=default

shindig.canonical.json.db=sampledata/canonicaldb.json
```

7) Create database "shindig" and give access to it for user "shindig" with password "shindig".

8) Compile now the project and run jetty server according to 2) The tables will be created in the database based on shindig's default database structure.

9) When database is created, add manually a row to the Person table with oid = 1, person_id = 1 and display_name = Shindig User

10) Run the jetty server as explained in 2) (Ignore errors that tables already exist). Now execute a social request in your browser.

```
http://localhost:8080/social/rest/people/1/@self
```

You should get back a json response (I had to reload page 3 times to get it)

```
{"entry":{"ims":[],"applcitions":[],"emails":[],"addresses":[],"filterCapability":{},"networkPresence":{"value":"XA","displayValue":"Extended Away"},"id":"1","objectId":1,"accounts":[],"urls":[],"organizations":[],"phoneNumbers":[],"photos":[],"displayName":"Shindig User"}}
```

This is a proof-of-concept tutorial on how to connect Shindig to a real MySQL database. Next tutorial will explain how to change shindig classes to connect to your own database structure and not the one provided by shindig samples!

There is a patch as attachment to this document that makes all the discussed changes (please use it with care because shindig trunk might have changed when you read this and do not forget to change the absolute path to your own shindig signing keys)